**Practicals AI**

**QUESTION 1.**

**Write a prolog program to calculate the sum of two numbers.**

CODE :

sum(X, Y, Z) :- Z is X + Y.

:- initialization(main).

main :-

write('Enter first number: '),

read(X),

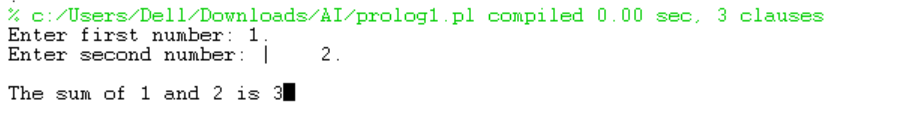
write('Enter second number: '),

read(Y),

sum(X, Y, Z),

write('The sum of '), write(X), write(' and '), write(Y), write(' is '), write(Z).

OUTPUT :



**QUESTION 2.**

**Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.**

**CODE :**

max(X, Y, M) :-

X>=Y,

M is X.

max(X, Y, M) :-

X<Y,

M is Y.

:- initialization(main).

main :-

write('Enter first number: '),

read(X),

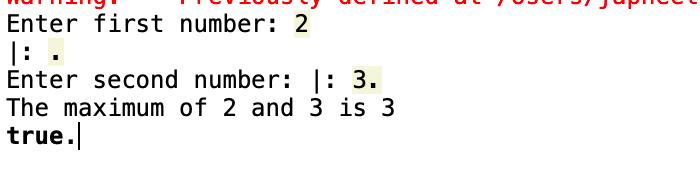
write('Enter second number: '),

read(Y),

max(X, Y, M),

write('The maximum of '), write(X), write(' and '), write(Y), write(' is '), write(M).

**OUTPUT :**



3



**QUESTION 3.**

**Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.**

**CODE :**

factorial(0, 1).

factorial(N, F) :- N > 0, N1 is N - 1, factorial(N1, F1), F is N \* F1.

:- initialization(main).

main :-

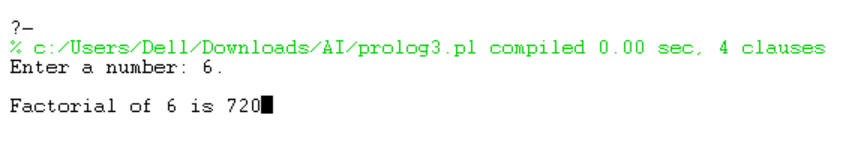
write('Enter a number: '),

read(N),

factorial(N, F),

write('Factorial of '), write(N), write(' is '), write(F).

**OUTPUT :**





**QUESTION 4.**

**Write a program in PROLOG to implement generate\_fib(N,T) where T represents the Nth term of the fibonacci series.**

**CODE :**

generate\_fib(0, 0).

generate\_fib(1, 1).

generate\_fib(N, T) :-

N>1,

N1 is N - 1,

N2 is N - 2,

generate\_fib(N1, T1),

generate\_fib(N2, T2),

T is T1 + T2.

:- initialization(main).

main :-

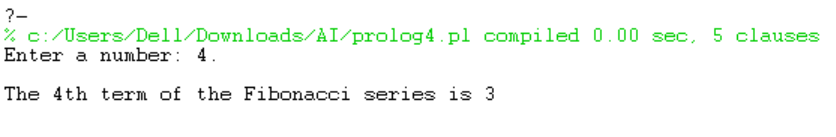
write('Enter a number: '),

read(N),

generate\_fib(N, T),

write('The '), write(N), write('th term of the Fibonacci series is '), write(T).

**OUTPUT :**





**QUESTION 5.**

**Write a Prolog program to implement GCD of two numbers.**

**CODE :**

gcd(X, Y, G) :-

X =:= Y,

G is X.

gcd(X, Y, G) :-

X<Y,

Y1 is Y - X,

gcd(X, Y1, G).

gcd(X, Y, G) :-

X>Y,

gcd(Y, X, G).

OUTPUT :



**QUESTION 6.**

**Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.**

**CODE :**

power(\_, 0, 1).

power(Num, Pow, Ans):- Pow>0, Pow1 is Pow-1, power(Num, Pow1, Ans1), Ans is Num \* Ans1.

**OUTPUT :**



**QUESTION 7.**

**Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.**

**CODE :**

multi(N1, N2, R) :- R is N1 \* N2.

**OUTPUT :**



**QUESTION 8.**

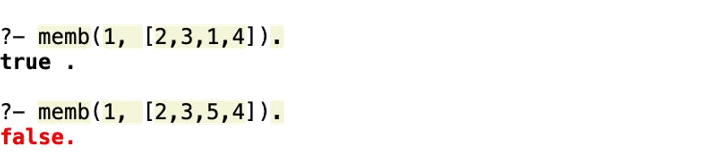
**Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.**

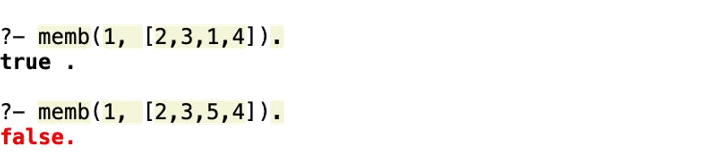
**CODE :**

memb(H,[H|\_]).

memb(H, [\_|T]) :- memb(H,T).

**OUTPUT :**





**QUESTION 9.**

**Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.**

**CODE :**

%concatenate(L1, L2, R)

concatenate([],[],[]).

concatenate([],L,L).

concatenate(L,[],L).

concatenate([H1|T1], L2, [H1|R]):- concatenate(T1, L2, R).

**OUTPUT :**





**QUESTION 10.**

**Write a Prolog program to implement reverse (L, R) where List L is original and List R is reversed list.**

**CODE :**

%reverse

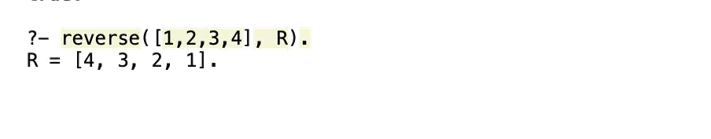
reverse([], []).

reverse([H|T], Reversed) :-

reverse(T, ReversedT),

append(ReversedT, [H], Reversed).

**OUTPUT :**





**QUESTION 11.**

**Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.**

**CODE :**

%reverse

%concatenate(L1, L2, R)

concatenate([],[],[]).

concatenate([],L,L).

concatenate(L,[],L).

concatenate([H1|T1], L2, [H1|R]):- concatenate(T1, L2, R).

reverse\_list([], []).

reverse\_list([H|T], R) :-

reverse\_list(T, R1),

concatenate(R1, [H], R).

palindrome(List) :-

reverse\_list(List, List).

**OUPUT :**



**QUESTION 12.**

**Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.**

**CODE :**

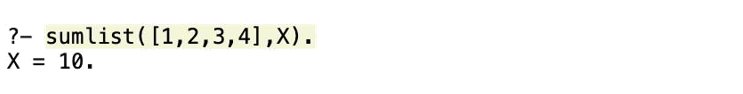
sumlist([], 0). % The sum of an empty list is 0.

sumlist([H|T], S) :-

sumlist(T, S1),

S is H + S1.

**OUTPUT :**





**QUESTION 13.**

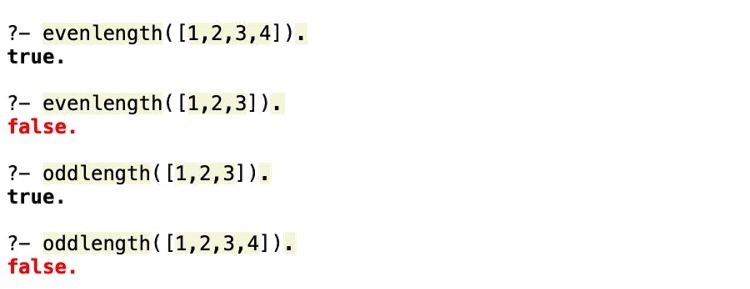
**Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.**

**CODE :**

evenlength(List) :- length(List, N), 0 is N mod 2.

oddlength(List) :- length(List, N), 1 is N mod 2.

**OUTPUT :**





**QUESTION 14.**

**Write a Prolog program to implement nth\_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.**

**CODE :**

nth\_element(1, [X|\_], X).

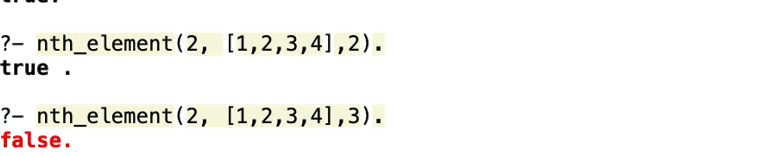
nth\_element(N, [\_|T], X) :-

N>1,

N1 is N - 1,

nth\_element(N1, T, X).

**OUTPUT :**





**QUESTION 15.**

**Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.**

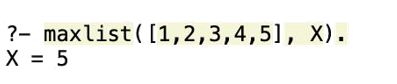
**CODE :**

maxlist([X], X).

maxlist([H|T], M) :-

maxlist(T, M1), M is max(H, M1).

**OUTPUT :**





**QUESTION 16.**

**Write a prolog program to implement insert\_nth (I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.**

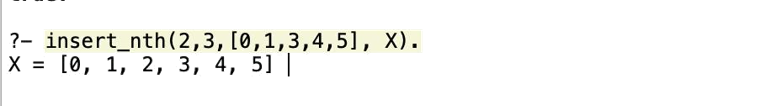
**CODE :**

insert\_nth(I, 1, [], [I]).

insert\_nth(I, 1, L, [I|L]).

insert\_nth(I, N, [H|T], [H|R]) :- N > 1, N1 is N - 1, insert\_nth(I, N1, T, R).

**OUTPUT :**



**QUESTION 17.**

**Write a Prolog program to implement delete\_nth (N, L, R) that removes the element on Nth position from a list L to generate a list R.**

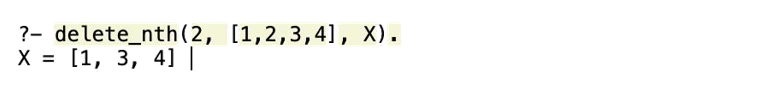
**CODE :**

delete\_nth(\_, [], []).

delete\_nth(1, [\_|T], T).

delete\_nth(N, [H|T], [H|R]) :- N > 1, N1 is N - 1, delete\_nth(N1, T, R).

**OUTPUT :**





**QUESTION 18.**

**Write a program in PROLOG to implement merge (L1, L2, L3) where L1 is first ordered list and L2 is second ordered list and L3 represents the merged list.**

**CODE :**

merge([], L, L).

merge(L, [], L).

merge([H1|T1], [H2|T2], [H1|T3]) :-

H1 =< H2,

!,

merge(T1, [H2|T2], T3).

merge([H1|T1], [H2|T2], [H2|T3]) :-

merge([H1|T1], T2, T3).

**OUTPUT :**

